###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студента 2 курса, группы 21209

Усольцева Антона Андреевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

М.А.Мичуров

В.А.Перепёлкин

Новосибирск 2022

**СОДЕРЖАНИЕ**

Цель 3

Задачи 3

Описание работы 4

Заключение 8

# ЦЕЛЬ

1. Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

# ЗАДАЧИ

1. Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя (Vendor ID) и идентификатор изделия (Product ID).

2. Изучить состав и характеристики обнаруженных с помощью реализованной программ USB устройств.

3. Дополнить программу, реализованную в п. 2 функцией печати серийного номера USB устройства. Для написания функции рекомендуется использовать функции libusb\_open, libusb\_close, libusb\_get\_string\_descriptor\_ascii для печати поля iSerialNumber дескриптора устройства.

# ОПИСАНИЕ РАБОТЫ

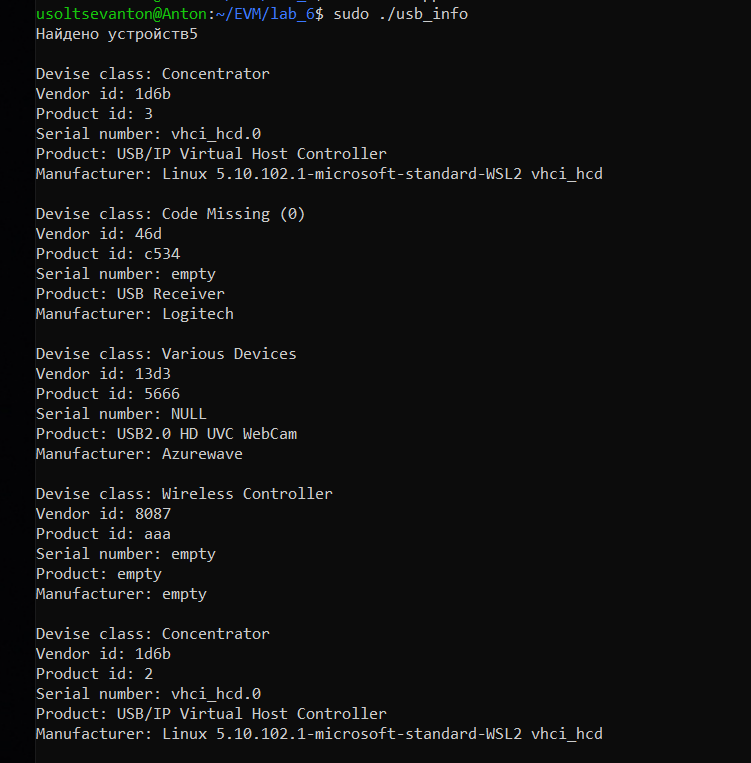
Листинг 1

#include <iostream>  
#include <libusb-1.0/libusb.h>  
#include <map>  
  
using namespace std;  
  
void printdev(libusb\_device \*dev);  
  
int main() {  
 libusb\_device \*\*devs; // указатель на указатель на устройство,  
 // используется для получения списка устройств  
 libusb\_context \*ctx = NULL; // контекст сессии libusb  
 int r; // для возвращаемых значений  
 ssize\_t cnt; // число найденных USB-устройств  
  
 // инициализировать библиотеку libusb, открыть сессию работы с libusb  
 r = libusb\_init(&ctx);  
 if (r < 0) {  
 cerr << "Ошибка: инициализация не выполнена, код:\n" << r << endl;  
 return 1;  
 }  
 // получить список всех найденных USB- устройств  
 cnt = libusb\_get\_device\_list(ctx, &devs);  
 if (cnt < 0) {  
 cerr << "Ошибка: список USB устройств не получен.\n" << endl;  
 return 1;  
 }  
 cout << "Найдено устройств " << cnt << "\n" << endl;  
 ssize\_t i; // индексная переменная цикла перебора всех устройств  
 for (i = 0; i < cnt; i++) // цикл перебора всех устройств  
 printdev(devs[i]); // печать параметров устройства  
  
 // освободить память, выделенную функцией получения списка устройств  
 libusb\_free\_device\_list(devs, 1);  
 libusb\_exit(ctx); // завершить работу с библиотекой libusb,  
 // закрыть сессию работы с libusb  
 return 0;  
}  
void printdev(libusb\_device \*dev) {  
 map<int, string> classes = { //основные классы устройств  
 {0, "Code Missing (0)"},  
 {1, "Audio device"},  
 {2, "Network adapter"},  
 {3, "User Interface Device"},  
 {5, "Physical Device"},  
 {6, "Images"},  
 {7, "Printer"},  
 {8, "Storage"},  
 {9, "Concentrator"},  
 {10, "CDC-Data"},  
 {11, "Smart Card"},  
 {13, "Content Security"},  
 {14, "Video device"},  
 {15, "Personal Medical Device"},  
 {16, "Audio and video devices"},  
 {220, "Diagnostic Device"},  
 {224, "Wireless Controller"},  
 {239, "Various Devices"},  
 {254, "Specific device"},  
 };  
 libusb\_device\_descriptor desc; // дескриптор устройства  
 int r = libusb\_get\_device\_descriptor(dev, &desc);  
 if (r < 0) {  
 cerr << "Ошибка: дескриптор устройства не получен, код:\n" << r << endl;  
 return;  
 }  
 int deviceClass = static\_cast<int>(desc.bDeviceClass);  
 if(classes.count(deviceClass)>0)  
 cout << "Devise class: " << classes[deviceClass] << endl;  
 else  
 cout << "Devise class is unknown" << endl;

cout << "Vendor id: " << hex << desc.idVendor << endl;  
 cout << "Product id: " << desc.idProduct << endl;  
  
 libusb\_device\_handle \*handle = nullptr;  
 if (libusb\_open(dev, &handle) == LIBUSB\_SUCCESS) {  
 unsigned char str[256];  
 r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iSerialNumber, str, sizeof(str));  
 if (r > 0)  
 cout << "Serial number: " << str << endl;  
 else  
 cout << "Serial number: empty" << endl;  
  
 r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iProduct, str, sizeof(str));  
 if (r > 0)  
 cout << "Product: " << str << endl;  
 else  
 cout << "Product: empty" << endl;  
  
 r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iManufacturer, str, sizeof(str));  
 if (r > 0)  
 cout << "Manufacturer: " << str << endl;  
 else  
 cout << "Manufacturer: empty" << endl;  
 } else  
 cerr << "FAILURE" << endl;  
 cout << endl;  
 libusb\_close(handle);  
}

Команда компиляции: g++ -o usb\_info /usr/include/libusb-1.0/libusb.h usb.cpp -lusb-1.0

Результат исполнения:



Программа выводит количество обнаруженных USB устройств и для каждого устройства выводит:

1. его класс
2. идентификатор производителя
3. идентификатор изделия
4. серийный номер
5. тип (имя) устройства
6. фирму производителя

Выводит эти данные программа на основе дескриптора устройства (если что-то не удалось выяснить из дескриптора устройства, то в остальные дескрипторы (конфигурации, интерфейса) программа не смотрит).

Структура дескриптора устройства представлена ниже

Общий вид дескриптора устройства

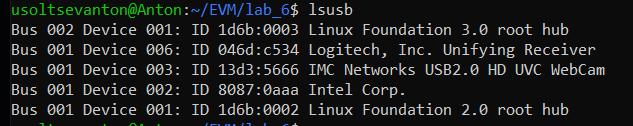
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | bLength | 1 | Число | Размер дескриптора в байтах (для дескриптора устройства размер 18 байт) |
| 1 | bDescriptorType | 1 | Константа | Тип - Device Descriptor (0x01) |
| 2 | bcdUSB | 2 | BCD | Номер спецификации USB, с которой совместимо устройство. |
| 4 | bDeviceClass | 1 | Class (класс) | Код класса (назначается организацией USB Org) Если равно 0, то каждый интерфейс указывает свой собственный код класса. Если равен 0xFF, то код класса определяется вендором. Иначе поле содержит код стандартного класса. |
| 5 | bDeviceSubClass | 1 | SubClass (подкласс) | Код подкласса (назначается организацией USB Org) |
| 6 | bDeviceProtocol | 1 | Протокол | Код протокола (назначается организацией USB Org) |
| 7 | bMaxPacketSize | 1 | Число | Максимальный размер пакета для конечной точки 0. Допустимые размеры 8, 16, 32, 64 |
| 8 | idVendor | 2 | ID | Vendor ID, VID (назначается организацией USB Org) |
| 10 | idProduct | 2 | ID | Product ID, PID (назначается организацией - производителем) |
| 12 | bcdDevice | 2 | BCD | Device Release Number (номер версии устройства) |
| 14 | iManufacturer | 1 | Индекс | Индекс строки, описывающей производителя |
| 15 | iProduct | 1 | Индекс | Индекс строки, описывающей продукт |
| 16 | iSerialNumber | 1 | Индекс | Индекс строки, содержащей серийный номер |
| 17 | bNumConfigurations | 1 | Целое (Integer) | Количество возможных конфигураций |

Идентификатор производителя (idVendor) и идентификатор изделия (idProduct) выводятся напрямую, как элементы структуры desc (Листинг 1)

По параметру bDeviceClass ищется соответствующий класс устройства в map

По параметрам (param) iManufacturer, iProduct, iSerialNumber ищутся оставшиеся данные. Для этого используется функция libusb\_get\_string\_descriptor\_ascii(handle, desc.param, str, sizeof(str));

Также проверим корректность некоторых найденных данных (идентификатор производителя, идентификатор изделия) функцией библиотеки libusb -- lsusb



# ЗАКЛЮЧЕНИЕ

В ходе работы была изучена библиотека libusb, которая реализует универсальный низкоуровневый интерфейс, который позволяет взаимодействовать со всеми возможными USB-устройствами. С использованием методов этой библиотеки была написана программа, которая получает список доступных USB-устройств и узнает их параметры. Также была изучена некоторая структура дескрипторов устройства.